Classificador de transações por cartão de crédito

Andriel Carlos de Souza Biagioni
RA - 189935
andriel.biagioni@gmail.com

Ricardo Bernardini Gonçales
RA - 189844
ricardobgoncales@gmail.com

***Resumo*: Nos últimos anos, muito dinheiro tem sido perdido devido a fraudes por cartão de crédito, forçando com que novas medidas de segurança sejam adotadas pelas instituições financeiras. A eficiência dos algoritmos utilizados é fundamental para reduzir esses prejuízos, e novos estudos indicam que técnicas avançadas de aprendizado de máquinas são a chave para o sucesso das investigações dessas fraudes. Em vista desse desafio, este trabalho propõe explorar um conjunto de dados real no que concerne a transações por cartão de crédito fornecido por uma grande empresa europeia, e avaliar diferentes modelos de classificação sobre esses dados, a fim de averiguar a relação dos diferentes recursos coletados com a legitimidade de cada transação.**

I. Introdução

O atual cenário financeiro mundial enfrenta uma crescente presença de iniciativas de fraude e busca por sistemas para implementar soluções para esse problema. Esses sistemas são essenciais, uma vez que nem sempre é possível para um analista humano detectar padrões fraudulentos em conjuntos de transações, muitas vezes caracterizados por uma grande quantidade de dados e muitas dimensões. Além disso, os fraudadores mudam constantemente suas estratégias para evitar de serem detectados, o que faz com que as ferramentas tradicionais de proteção contra golpes se tornem inadequadas. Por isso, o uso da aprendizagem de máquinas tem sido amplamente discutido como uma possível solução. Vários sistemas de detecção baseados em técnicas de aprendizado de máquina foram usados com sucesso nesse problema [1].

Ao construir um modelo de detecção de golpes em cartão de crédito, há vários fatores que têm um impacto importante durante a fase de treinamento: assimetria dos dados, sensibilidade ao custo da aplicação, resposta de curto prazo do sistema, dimensionalidade do conjunto de dados e como pré-processar os recursos. As ações tomadas contra golpes podem ser divididas em *prevenção*, que tenta bloquear transações fraudulentas durante sua execução, e *classificação*, onde as transações de fraude bem sucedidas são identificadas a posteriori. Este trabalho explorará um conjunto supervisionado de dados pré-processados de transações por cartões de créditos, a fim de se estudar a eficiência de diferentes modelos de classificadores disponíveis na literatura durante a verificação da legitimidade dessas transações, não levando em conta sua execução em tempo real.

Em um modelo de classificação de fraude através do uso de cartão de crédito, é muito importante salientar os recursos que permitem uma classificação precisa. Espera-se que o uso de informações de transações acumuladas ao longo do tempo, resulte em uma melhor discriminação de fraude do que se pode obter ao nível de transações isoladas. Entretanto, dada a natureza do conjunto de dados utilizado nesse trabalho, análises sob a perspectiva do tempo levaram em conta apenas o período de dois dias corridos, no qual os dados foram coletados.

II. Revisão Bibliográfica

Para o desenvolvimento desse trabalho, foi considerado o estudo das 3 melhores análises sobre o problema de detecção de fraude, disponível em [2].

Primeiramente, explorou-se o conteúdo do trabalho realizado por [3]. Nele foram aplicadas algumas técnicas para melhor explorar a base, como a normalização do recurso *Montante*, remoção do recurso *Tempo* - julgado pelo autor como sem relevância - e também a utilização da técnica de *Resample* [4] na base desbalanceada, a fim de originar um subconjunto de dados com mesma proporção de transações entre as classes.

A partir dessa abordagem inicial, o autor utilizou o método de Regressão Logística, com variações em seus parâmetros de configuração, para assim buscar a melhor estrutura capaz de classificar as classes de transações sobre a base de treino. A avaliação dos resultados foi realizada considerando o índice da métrica *Recall*. Uma vez verificada a configuração melhor avaliada, buscou-se analisar seu comportamento sob o conjunto de treino trabalhado com a técnica de *Resample*. Este, por sua vez, resultou em um *Recall* de 93%, levando a entender que o desbalanceamento das classes é uma característica relevante sob a perspectiva dos resultados.

Outra referência que permitiu um aprofundamento nos estudos realizados nesse trabalho é [5]. As técnicas exploradas pelo autor variam desde desconsiderar recursos cujas variâncias não são expressivas para a predição dos dados, até utilizar diferentes estruturas de Rede Neural por meio de APIs do *Tensorflow* [6]. Isso o permitiu alcançar um valor final de *Recall* igual a 88% em seus dados de teste.

Em [7], um estudo detalhado sobre aplicação de árvore de decisão na base de dados aqui trabalhada avaliou a eficiência de alguns algoritmos em tempo e seus respectivos valores de acurácia. Os modelos analisados foram *GBM*, *XGBoost* e *light-GBM*. Para cada um desses modelos foram realizadas análises

a medida que diferentes parâmetros iam sendo alterados, de modo a buscar o menor tempo de execução e o maior valor de acurácia. O modelo melhor avaliado levou em conta a técnica *XGBoost*.

Com base na conclusão desses estudos, esse trabalho reuniu algumas das principais técnicas de classificação com o objetivo de alcançar melhores resultados, dada a métrica de referência para a base de dados escolhida.

III. Análise e Exploração de Dados

Nesta seção, primeiro o conjunto de dados usado para as experiências é descrito. Posteriormente, a estratégia adotada para seu particionamento é apresentada.

A. Conjunto de dados

O conjunto de dados *Credit Card Fraud Detection*, atualmente disponível na plataforma *Kaggle* [2], foi coletado e analisado durante uma colaboração de pesquisa da *Worldline* e do Grupo de Aprendizagem de Máquinas da ULB (*Université Libre de Bruxelles*) sobre Mineração de Dados e Detecção de Fraude [8].

Table I
Mapeamento das transações de entrada

Tempo	Segundos desde a primeira transação	Não normalizado
V1 à V28	Variáveis numéricas anonimizadas	Normalizadas
Montante	Valor em dólares da transação	Não Normalizado
Classe	Variável de resposta	0: genuíno, 1: fraude

Ele consiste em 284.807 linhas de dados, cada uma correspondendo a uma transação por cartão de crédito que é composta por 28 recursos numéricos anonimizados e normalizados (obtidos a partir de um conjunto de recursos originais usando a análise de componentes principais ou PCA, um método padrão de normalização de recursos que será explicado abaixo), bem como dois recursos adicionais: o tempo que a transação ocorreu e seu valor em dólares.

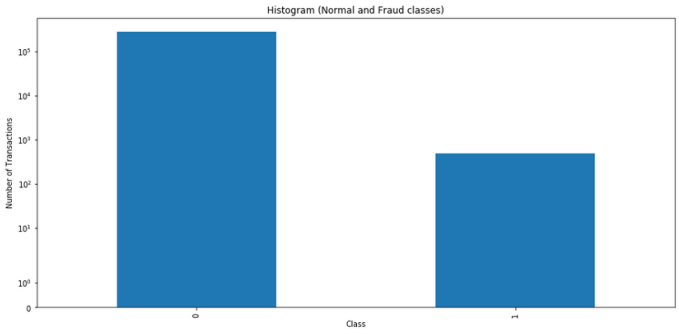

Figure 1. Histograma em escala *Symlog* das transações de entrada categorizadas em Normais (0) e Fraudes (1).

O recurso *Classe* é a variável de resposta e é necessário o valor 1 em caso de fraude ou 0, caso contrário. Uma característica muito importante a respeito dessas classes é seu desbalanceamento; As fraudes representam 0,172% de todas as transações (492 fraudes das 284.807 transações). Através da Figura 2, é possível notar o baixo índice de transações fraudulentas capturadas no período de 2 dias quando comparado com o índice de transações rotuladas genuínas. Por razões de confidencialidade, o significado da maioria das variáveis não foi revelado. O identificador do titular do cartão também não está disponível para que cada transação possa ser considerada independente das outras. Este é um dos raros conjuntos de dados sobre detecção de fraude disponíveis para a comunidade.

Os recursos normalizados compõem um vetor e são nomeadas de $V1$ à $V28$. Embora anonimizados, eles codificam vários aspectos da transação: normalmente a bandeira do cartão, o tipo da transação (por exemplo, pagamento, reembolso, solicitação de saldo), tipo de comerciante (quando aplicável, por exemplo, supermercado), canal (por exemplo, terminal *ATM* ou *POS*), modo de verificação (por exemplo, chip e *PIN*, faixa magnética), entre outros.

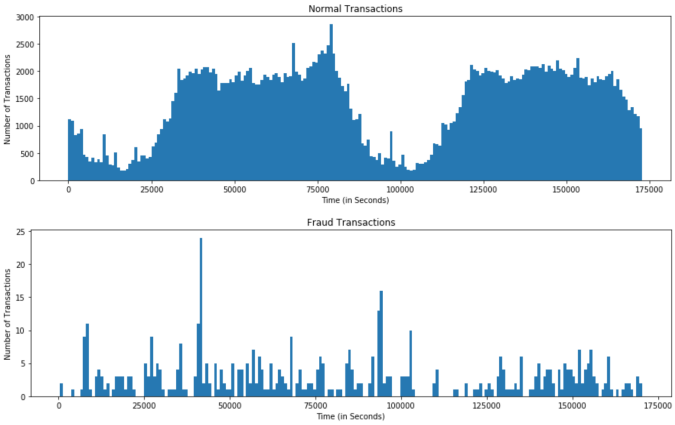

Figure 2. Gráfico do número de transações normais e fraudulentas, respectivamente, de acordo com o tempo.

1) PCA: : Os conjuntos de dados típicos para detecção de fraudes são geralmente grandes e multidimensionais. Dado que trabalhar com um menor número de variáveis pode permitir uma melhor compreensão do problema, estratégias para redução de dimensionalidade, como a de PCA, são geralmente aplicadas sobre esses dados. Em termos matemáticos, a PCA é uma técnica em que n variáveis aleatórias correlacionadas são transformadas em $d <= n$ variáveis não correlacionadas. As variáveis não correlacionadas são combinações lineares das variáveis originais e podem ser usadas para expressar os dados de forma reduzida. Normalmente, o primeiro componente principal da transformação é a combinação linear das variáveis originais com a maior variância. Em outras palavras, o primeiro componente principal é a projeção na direção em que a variância da projeção é maximizada. O segundo componente principal é a combinação linear das variáveis originais com a segunda maior variância e ortogonal ao primeiro componente principal, e assim por diante.

B. Particionamento dos dados

Antes de dar início a fase de treinamento dos modelos, foi realizada uma divisão do conjunto de dados em uma parte de

treinamento e outra de teste, cada uma delas correspondendo a 70% e 30% do conjunto total, respectivamente. É importante ressaltar que esse processo de divisão envolveu estratificação dos dados, permitindo com que a probabilidade de ver uma transação fraudulenta fosse aproximadamente a mesma tanto nos dados de treinamento como nos dados de teste.

Table II
PARTICIONAMENTO DOS DADOS

284.807	Total de transações
199.365	Total de transações no conjunto de treino
85.442	Total de transações no conjunto de teste

A parte de treino foi a mesma para todos os modelos avaliados. Ao final da comparação dos resultados, dada a métrica de avaliação, o melhor modelo foi selecionado e usado com a porção de dados disponíveis para o teste, sendo essa uma porção de amostras nunca antes usada. O processo de divisão dos dados entre treino e validação se deu através do método de validação cruzada *K-Fold*.

1) Validação cruzada K-Fold: No método de validação cruzada *K-Fold*, primeiramente, os dados são particionados em segmentos iguais (ou quase iguais). Em seguida, as iterações de treino e validação são realizadas de tal forma que, dentro de cada iteração, um seguimento diferente dos dados é escolhido para validação enquanto os demais seguimentos restantes são usados para treinamento.

Para melhor representação, os dados também são estratificados antes de serem divididos nesses seguimentos. A estratificação é o processo de rearranjo dos dados para garantir que cada seguimento seja um bom representante do todo. Isso pode ser alcançado através da definição do parâmetro *shuffle = True* do método *K-Fold*.

Um importante ponto a ser levantado é a determinação do valor de K. A ideia básica é que um K pequeno proporciona um custo menor ao modelo, mas que pode acabar proporcionando uma análise tendenciosa. Um K maior geralmente é mais custoso, menos tendencioso, mas pode sofrer grande variabilidade. Portanto, considerando o número de entradas do conjunto de dados desse trabalho e com base nos valores frequentemente citados na literatura, foi escolhido $K = 5$ para a confecção dos resultados.

IV. MÉTRICA DE AVALIAÇÃO

Um modelo de classificação é um mapeamento de instâncias para classes preditas. Utilizando o conjunto de dados disponível em [2], os modelos apresentados nesse trabalho produzem uma saída categórica em que diferentes limiares são aplicados para prever a adesão da classe predita.

Para avaliar a qualidade dos resultados obtidos, buscou-se verificar métricas como *Recall* e *F1 Score*.

Entretanto, os dados são altamente desbalanceados, o conjunto de entrada possui 492 transações classificadas como fraudes e 248.315 transações genuínas. Por conta dessa alta taxa de desbalanceamento, a *Recall* foi escolhida como a métrica estatística principal para determinar o melhor modelo. Vale ressaltar que os estudos desse problema disponibilizados na plataforma *Kaggle* também utilizam dessa métrica como referência.

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos} \quad (1)$$

$$Recall = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos} \quad (2)$$

$$F1\ Score = \frac{2 * (Recall * Precisão)}{Recall + Precisão} \quad (3)$$

Sendo assim, o melhor modelo será aquele que possuir o *Recall* mais alto, isto é, mais próximo de 1.

V. LINGUAGEM E MÓDULO DE DESENVOLVIMENTO

Para implementação dos modelos de classificação estudados, utilizou-se a linguagem de programação *Python* e o *Scitkit-Learn*, um poderoso módulo disponível para aprendizagem de máquinas. Esse módulo permite interagir com funções para regressão, classificadores, redução de dimensionalidade, dentre outros.

VI. CLASSIFICADORES

Diferentes modelos foram propostos a fim de se comparar várias abordagens de classificação. Ao utilizar uma variedade de modelos de classificação, espera-se poder tirar conclusões mais gerais sobre classificação de fraude, em vez de conclusões específicas apenas para um modelo particular. Deve-se lembrar, porém, que as configurações de parâmetros ótimos para qualquer método variam de um conjunto de dados para outro, por conta disso, o treinamento foi realizado em cima de dados estratificados, sempre considerando simular como eles se encontrariam no mundo real. Uma vez definidas as configurações, nada foi alterado para a fase de teste.

Trabalhar com vários modelos permitiu consolidar os diferentes pontos fortes e fracos de cada um, dado o problema estudado. Além disso, o fato dos autores possuírem conhecimentos razoáveis na utilização de todos os métodos utilizados ajuda a fornecer uma avaliação clara entre eles. Foi tomado o cuidado de investir aproximadamente a mesma quantidade de tempo e esforço humano no uso de cada uma dessas técnicas de classificação, no que diz respeito aos parâmetros de otimização. Mais uma vez, embora tal avaliação seja subjetiva, isso deve ajudar a tornar a comparação mais justa.

Geralmente, é verdade que o desempenho da maioria dos classificadores pode ser melhorado em qualquer caso, aplicando um esforço extra em parâmetros de ajuste fino ou modificando os dados. Neste estudo, fizemos apenas esforços rudimentares nesse sentido e tentamos garantir que nenhuma técnica fosse favorecida. Dessa forma, os resultados relatados aqui não representam o melhor desempenho alcançável por cada modelo com esses dados, embora acredita-se que eles estejam próximos dos esperados.

A. Regressão Logística

O primeiro modelo abordado nesse trabalho foi o de Regressão Logística. Esse é um tipo de regressão que prevê a probabilidade de ocorrência de um evento ao ajustar os dados a uma função logística. Para isso, ela faz uso de várias variáveis preditoras que podem ser numéricas ou categóricas. Por exemplo, a probabilidade de uma transação do conjunto *Credit Card Fraud Detection* ser ou não uma fraude pode ser predita a partir da análise de variáveis anônimas derivadas de um resultado PCA.

A aplicação tradicional do modelo de Regressão Logística é a classificação binária, uma vez que ela é baseada na função $sigmoide$ [9]. No entanto, usando abordagens como *One-vs-All* ou *One-vs-One* - através da Regressão Logística Multinomial - é possível também enfrentar problemas de classificação em várias classes, embora esse não seja o caso desse trabalho.

B. Rede Neural - Múltiplas Camadas

Redes Neurais fornecessem uma diferente estrutura para resolver problemas tradicionais de classificação, como o abordado nesse documento. Elas permitem definir uma camada de entrada, uma camada de saída - assim como já visto em Regressão Logística - e uma ou mais camadas ocultas. Os neurônios são organizados em camadas de tal forma que eles se mantêm conectados apenas a outros neurônios de uma camada diferente da sua. Cada entrada de um neurônio é multiplicada pelo seu peso e então somadas com o *Bias* para que, assim, se aplique a este valor uma função de ativação não linear de modo a determinar a saída de acordo com um valor limiar.

Nesse trabalho, explorou-se mais de uma configuração de Rede Neural, a fim de melhor compreender os diferentes efeitos proporcionados nos resultados a partir de alterações em sua estrutura. Assumiu-se, portanto, que as redes criadas deveriam variar o número de camadas ocultas - entre 2 a 4 -, o número de neurônios em cada uma dessas camadas ocultas, a função de ativação e a função para o cálculo do gradiente.

Uma vez que, em teoria, qualquer função diferencial pode ser usada como função de ativação, optou-se nesse trabalho por utilizar as funções mais comuns, como a *ReLU*, *Tanh* e a *Função Logística*, como também as funções para o cálculo do gradiente descendente *SGD*, *Adam* e *L-BFGS*.

C. XGBoost

O método XGBoost [11], conhecido como *Extreme Gradient Boosting*, implementa árvore de decisão utilizando técnicas já conhecidas em algoritmos como *SGB* e *GBM* [10]. O refinamento dessas duas técnicas, aliado ao alto poder de processamento paralelo e a escalabilidade para o *Hadoop* que o *XGBoost* possui, torna-o incrivelmente rápido em cálculos, na criação das árvores de decisão e fornece melhores predições quando comparado com *SGB* e *GBM*.

Através de parâmetros de configuração, como por exemplo o *max_depth*, é possível acelerar a predição por intermédio de podas na ramificação da árvore de decisão, de acordo com a configuração da máxima profundidade permitida. Já a declaração de pequenos valores para a taxa de aprendizado permite melhor generalizar os dados, tendendo para que os resultados, dado o conjunto de teste, apresentem melhores valores. O parâmetro *booster* auxilia na aprendizagem, melhorando a classificação dos dados em um conjunto desbalanceado.

Nesse trabalho foi utilizado o método *XGBClassifier*, disponível na biblioteca do *XGBoost*, para a classificação dos dados diante das classes de transação (Fraude e Normal).

VII. Experimentos e Discussões

Os experimentos com todos os modelos foram executados utilizando a base dividida aleatoriamente em uma proporção de 70% para os dados de treino/validação e 30% para teste. Considerando que cada transação possui um total de 30 recursos (desconsiderando seu rótulo), optou-se por trabalhar com apenas 28 deles, os quais foram disponíveis após uma transformação PCA. Essa escolha surgiu como consequência de comparações entre nossos experimentos, pois em todos eles, ao se levar em conta os recursos *Tempo* e *Montante*, os resultados obtidos se encontravam inferiores.

A utilização da técnica de *K-Fold* foi fundamental para evitar *overfitting*. Embora a escolha de $K = 5$ foi a mesma em todos os experimentos, dado o desbalanceamento das classes, foi possível observar que os resultados sofriam consideráveis variações mesmo quando executados com um mesmo modelo sem alterações. Mesmo levando em conta a estratificação dos dados, esse foi um comportamento comum na fase de treinamento, e acredita-se que isso também seja observado em casos reais.

No primeiro experimento, foi explorado o modelo de Regressão Logística com variações em seus parâmetros de configuração, como as estratégias de multi-classes (*ovr* e *multinomial*), funções de otimização para convergência do resultado e também o parâmetro de regularização. O modelo que obteve o melhor resultado considerando o conjunto de treinamento foi o configurado da seguinte maneira:

```
Best Model for LogisticRegression:

C=1, class_weight=None,
dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100,
multi_class='ovr', n_jobs=1,
penalty='l2',solver='liblinear',
tol=0.0001, warm_start=False
```

Para o modelo de Rede Neural foi escolhido o MLP *Multi-Layer Perceptron*. Sua estrutura é composta por camadas escondidas que realizam a retro-propagação. Nesse trabalho, buscou-se variar a quantidade de camadas escondidas - 2 a 4 camadas -, a quantidade de neurônios por camada - 10 a 49 neurônios -, função de ativação e também a função para o cálculo do gradiente. Foi mantido o número de iterações pois, em análises preliminares foi possível verificar que o modelo teve poucas variações em seus resultados, dada a mesma base.

Após a verificação da estrutura de Rede Neural melhor avaliada sob a base de treino, foi realizada uma abordagem

especial para um experimento, levando em conta o estudo realizado por [2], onde se utiliza da técnica de *Resample* para obter um subconjunto de dados balanceado e retestar o modelo sobre essa nova base. Entretanto, o resultado obtido utilizando esse subconjunto balanceado não foi bom, diminuindo bastante o valor de *Recall* alcançado.

O melhor modelo estruturado com Rede Neural considera o uso de 2 camadas escondidas com 15 e 27 neurônios respectivamente. A função de ativação melhor avaliada foi a *Logística*, fazendo uso do método *Adam* para o cálculo do gradiente descendente.

```
Best Model for MLPClassifier:

activation= 'logistic', alpha=0.0001,
batch_size='auto', beta_1=0.9,
beta_2=0.999, early_stopping=False,
epsilon=1e-08, hidden_layer_sizes=(15, 27),
learning_rate='constant',
learning_rate_init=0.001,
max_iter=1000, momentum=0.9,
nesterovs_momentum=True, power_t=0.5,
shuffle=True, solver='adam',
tol=0.0001, validation_fraction=0.1,
verbose=False, warm_start=False
```

A escolha do *XGBoost* como outra abordagem para o problema se deve principalmente ao trabalho de [7], o qual demonstrou ser possível obter bons resultados ao se utilizar dessa técnica para essa base de dados. Em nosso trabalho, buscou-se explorar o método de classificação *XGBClassifier* do *XGBoost* com diferentes variações de parâmetro em sua configuração. Dentre as 27 configurações exploradas, a que obteve o melhor resultado durante o treinamento foi o modelo cuja profundidade máxima permitida na árvore é igual a 5, fazendo uso da função de *dart* para melhorar a convergência da classificação e uma taxa de aprendizagem de 0,1.

```
Best Model for XGBClassifier:

(base_score=0.5, booster= dart,
colsample_bylevel=1,gamma=0,
colsample_bytree=1, max_depth=5,
learning_rate=0.1, max_delta_step=0,
min_child_weight=1, subsample=1
missing=None, n_estimators=100,n_jobs=1,
nthread=4, objective='binary:logistic',
random_state=1,reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, seed=None,
silent=True
```

A tabela III destaca os valores obtidos para a métrica *Recall* resultantes dos treinamentos realizados com cada um dos melhores modelos citados anteriormente.

Table III
MELHORES RESULTADOS OBTIDOS DENTRE OS CLASSIFICADORES

Modelo	Recall
Regressão Logística	0,62108
Rede Neural (MLP)	0,79494
XGBClassifier (XGBoost)	0,78370

Desse modo, o modelo vencedor foi aquele que obteve o maior valor pra métrica *Recall*, sendo a Rede Neural o melhor entre os treinados nas configurações destacadas anteriormente.

Após a escolha do melhor modelo, dada a métrica *Recall*, aplicou-se sobre ele o conjunto de teste pela primeira vez. Ao final da execução do teste, o modelo avaliado obteve um *Recall* de *0,80882*, sendo esse um valor mais alto do que o obtido por ele durante a avaliação do treinamento.

Para compreender o resultado final, explorou-se a matriz de confusão a fim de visualizar a classificação real e predita, de acordo com as respectivas classes de transação (Normal e Fraude).

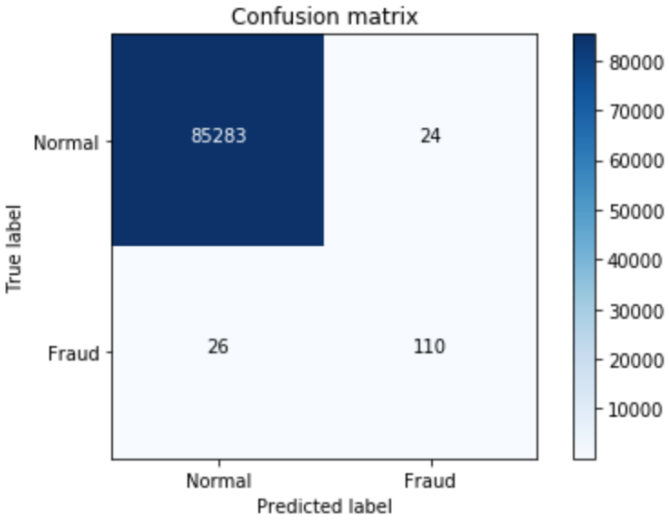

Figure 3.

A figura 3 resume o desempenho do classificador vencedor a partir do conjunto de teste. Os resultados são apresentados como uma matriz de confusão, onde as linhas correspondem à classe verdadeira e as colunas correspondem à classe estimada, como descritas na Seção III. Os valores diagonais da matriz de confusão fornecem a classe corretamente prevista, enquanto os valores não diagonais mostram a porcentagem de erro de rotulagem para cada classe.

O conjunto de teste é composto por um total de 136 transações fraudulentas, sendo que 19% delas foram classificadas erroneamente. É importante ressaltar que existem moderadamente mais transações rotuladas fraudulentas no conjunto de treinamento do que no conjunto de teste, com uma taxa proporcional de 0,18% de fraudes no conjunto de treino e 0,16% no conjunto de teste. Isso ajuda a explicar o resultado ser melhor quando realizada a execução do modelo sobre o conjunto de teste.

VIII. CONCLUSÃO E TRABALHOS FUTUROS

A detecção de fraude em transações de cartão de crédito é um dos domínios mais explorados no mercado anti-fraude e depende da análise automática de transações registradas para detectar comportamentos fraudulentos. Cada vez que um cartão de crédito é usado, os dados da transação, compostos

por uma série de atributos, são armazenados nos bancos de dados do provedor do serviço.

É importante lembrar que os algoritmos estudados foram aplicados em dados reais obtidos durante um curto período de coleta. Em casos reais, uma única informação de transação, normalmente, não é suficiente para detectar uma ocorrência de fraude e a análise deve considerar medidas agregadas como o total gasto por dia, o número de transação por semana ou o valor médio de uma transação [12]. No entanto, análises sob o ponto de vista de medidas agregadas não foram possíveis, uma vez que o conjunto de dados trabalhado foi coletado durante dois dias corridos em setembro de 2013, isto é, por um curto período.

Após uma comparação de três algoritmos diferentes, Regressão Logística, *XGBoost* e Rede Neural, em diferentes configurações, pode-se confirmar por intermédio desse trabalho que para os nossos dados, o modelo que melhor classificou as transações de entrada foi a Rede Neural (MLP) com 2 camadas escondidas, tendo cada uma delas 15 e 27 neurônios respectivamente.

Também se pode notar que a variação de K (número de vezes para a validação cruzada na fase de treinamento) não afeta realmente os resultados: esse método, de fato, tem, em geral, mais importância quando a amostra é pequena.

Dessa forma, os resultados obtidos nesse estudo constatam que poderá ser possível, em um futuro, a utilização de técnicas como a Rede Neural em sistemas de classificação de fraudes como as de cartão de crédito.

Como trabalho futuro sobre a abordagem de Rede Neural para esse problema, é possível considerar a utilização de outra ferramenta, como o *TensorFlow*, que dispõe de uma extensa lista de funções de ativação, funções para o cálculo de gradiente e também de técnicas para a rede que ainda não estão implementadas no módulo *Scikit-Learn*. Outro ponto a se considerar como trabalho futuro é a utilização da técnica de *drop-out* de neurônios em camadas, que não foi explorada nesse trabalho devido a limitações do módulo adotado para confecção do código.

REFERENCES

[1] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *http://www.sciencedirect.com/science/article/pii/S0167923610001326*

[2] "Credit Card Fraud Detection". *https://www.kaggle.com/dalpozz/creditcardfraud*

[3] joparga3 - In depth skewed data classif. *https://www.kaggle.com/joparga3/in-depth-skewed-data-classif-93-recall-acc-now*

[4] Kish, L. and Frankel M.R. (1974). Inference from complex samples. Journal of the Royal Statistical Society, Series B. Vol. 36, 1, pp. 1–37.

[5] Currie32 - Predicting Fraud with TensorFlow *https://www.kaggle.com/currie32/predicting-fraud-with-tensorflow*

[6] Google Research. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *http://download.tensorflow.org/paper/whitepaper2015.pdf*

[7] 'nschneider - GBM vs xgboost vs lightGBM' *https://www.kaggle.com/nschneider/gbm-vs-xgboost-vs-lightgbm*

[8] Machine Learning Group, ULB - "BruFence: Scalable machine learning for automating defense system". Available at *http://mlg.ulb.ac.be/BruFence*

[9] Wikipedia. Sigmoid Function. *https://en.wikipedia.org/wiki/Sigmoid_function*

[10] Introduction to Boosted Trees. *http://xgboost.readthedocs.io/en/latest/model.html*

[11] Friedman, Jerome H., Stochastic gradient boosting (1999)

[12] Hand DJ (2006) Classifier technology and the illusion of progress (with discussion). Available at *https://projecteuclid.org/download/pdfview_1/euclid.ss/1149600839*